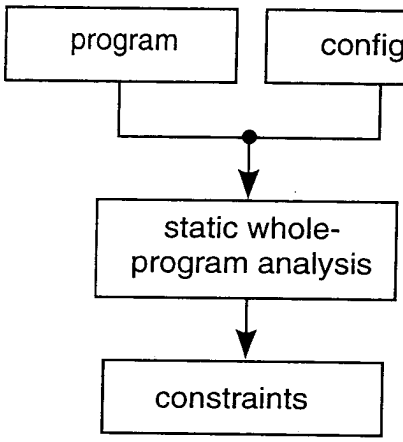

FIG. 1A

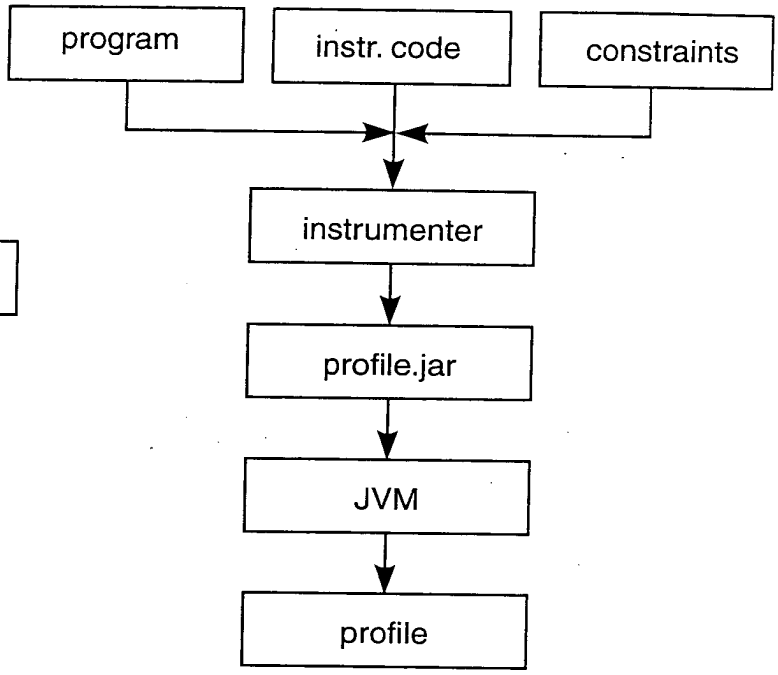

FIG. 1B

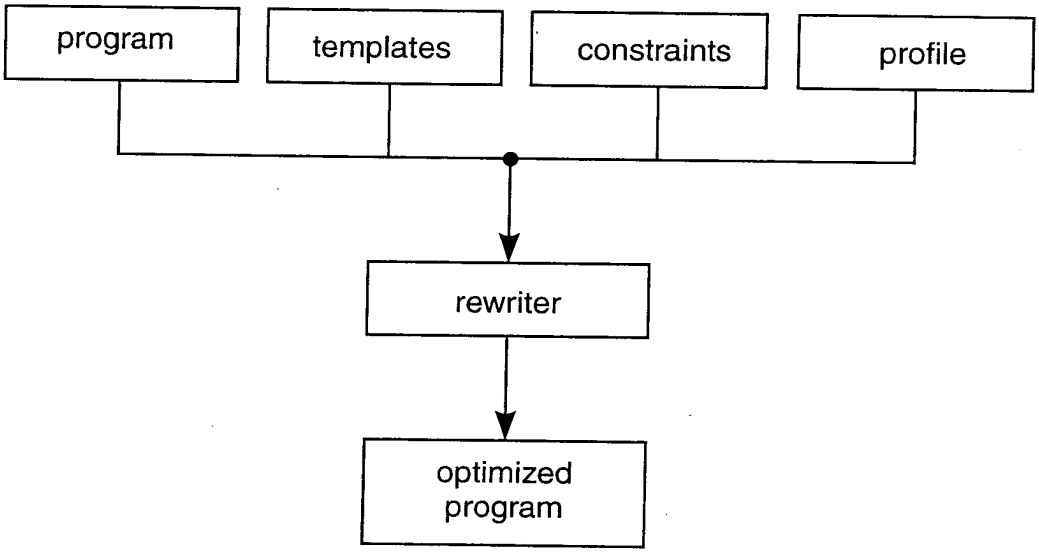

FIG. 1C

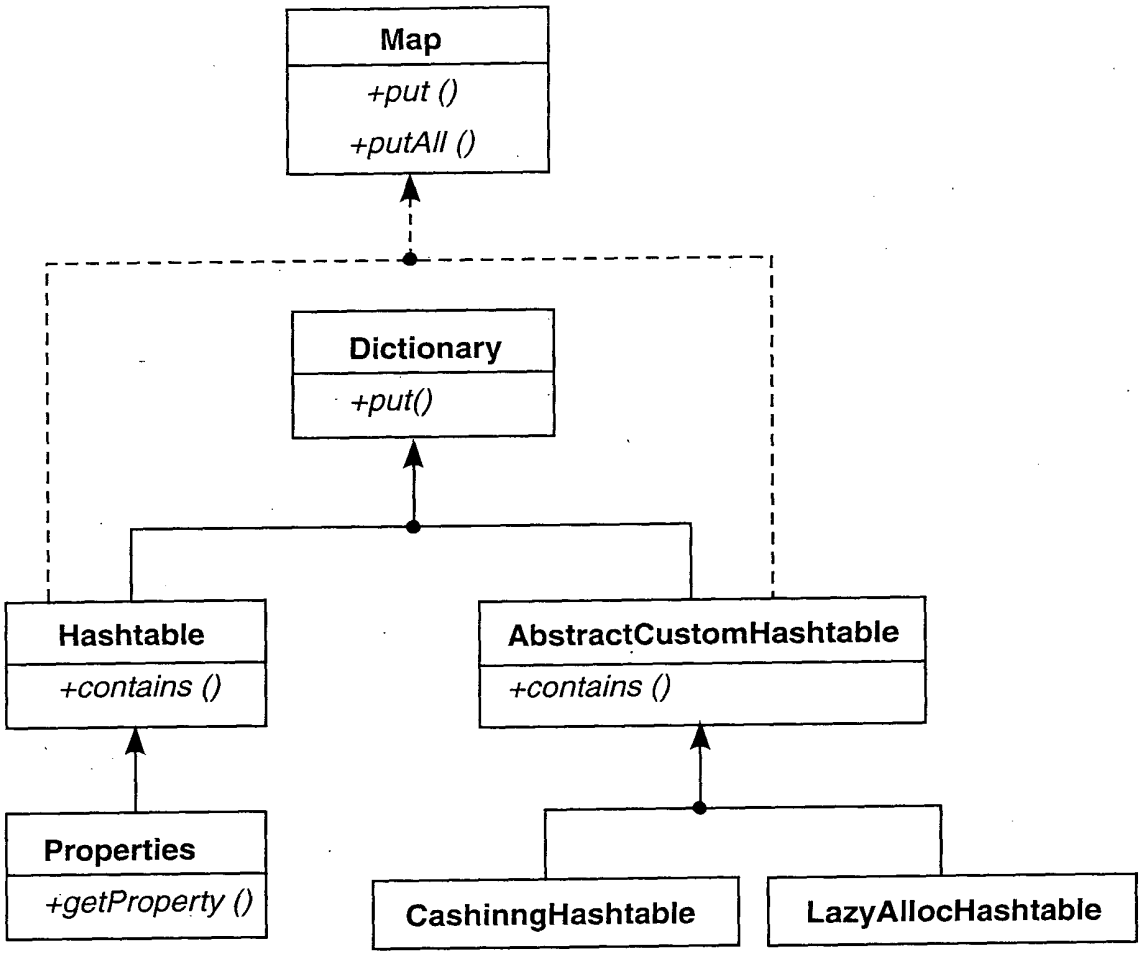
Map
+put ()
+putAll ()
Dictionary
+put()
Hashtable
+contains ()
AbstractCustomHashtable
+contains ()
Properties
+getProperty ()
CashinngHashtable
LazyAllocHashtable

FIG. 2

```
1  public class Example {
2      public static void foo(Map m) {
3          Hashtable h1 = new Hashtable( );     /* H1 */
4          JTree tree = new JTree(h1);
5          Hashtable h2 = new Hashtable( );     /* H2 */
6          Hashtable h3 = new Hashtable( );     /* H3 */
7          bar(h3);
8          h2 = h3;
9          h2.put("FOO", "BAR");
10         h2.putAll(m);
11         Properties p1 = System.getProperties();
12         String s = p1.getProperty("java.class.path");
13         h2 = p1;
14     }
15     public static void bar(Object o) {
16         Hashtable h4 = (Hashtable) o;     /* C1 */
17         if (h4.contains("FOO")){... }
18     }
19     public static void bad(){
20         String s = new String("bad");      /* S1 */
21         bar(s);
22     }
23 }
```

FIG. 3A

```
public class Example {
    public static void foo(Map m) {
        Hashtable h1 = new Hashtable( );
        JTree tree = new JTree(h1);
        Map h2 = new CachingHashtable( );
        Map h3 = new CachingHashtable( );
        bar(h3);
        h2 = h3;
        h2.put("FOO", "BAR");
        h2.putAll(m);
        Properties p1 = System.getProperties( );
        String s = p1.getProperty("java.class.path");
        h2 = p1;
    }
    public static void bar(Object o) {
        AbstractCustomHashtable h4 = (AbstractCustomHashtable) o;
        if (h4.contains("FOO")){... }
    }
    public static void bad( ) {
        String s = new String("bad");
        bar(s);
    }
}
```

FIG. 3B

[E]	the type of expression or *E*
[M]	the declared return type of method *M*
[F]	the declared type of .eld *F*
Decl(M)	the type that contains method *M*
Decl(F)	the type that contains .eld *F*
Param(M, i)	the *i*-th formal parameter of method *M*
$T' \leq T$	T' is equal to T, or T is a subtype of T
$T' < T$	T' is a proper subtype of T (i.e., $T' \leq T$ and not $T \leq T'$)

FIG. 4

program construct(s)/analysis fact(s)	implied type constraint(s)	
assignment $E_1 = E_2$	$[E_2] \leq [E_1]$	(1)
method call $E.m(E_1, \cdots, E_n)$ to a virtual method M	$[E.m(E_1, \cdots, E_n)] \triangleq [M]$ $[E_i] \leq [Param(M,i)]$ $[E] \leq Decl(M_1)$ or $\cdots$ or $[E] \leq Decl(M_k)$ where $RootDefs(M) = \{M_1, \cdots, M_k\}$	(2) (3) (4)
access $E.f$ to field F	$[E.f] \triangleq [F]$ $[E] \leq Decl(F)$	(5) (6)
return E in method M	$[E] \leq [M]$	(7)
constructor call new $C(E_1, \cdots, E_n)$ to constructor M	$[E_i] \leq [Param(M,i)]$	(8)
direct call $E.m(E_1, \cdots, E_n)$ to method M	$[E.m(E_1, \cdots, E_n)] \triangleq [M]$ $[E_i] \leq [Param(M,i)]$ $[E] \leq Decl(M)$	(9) (10) (11)
cast $(C)E$	$[(C)E] \leq [E]$ if $[E]$ is a class	(12)
for every type T	$T \leq$ java.lang.Object $[null] \leq T$	(13) (14)
implicit declaration of this in method M	$[this] \triangleq Decl(M)$	(15)
declaration of method M (declared in type T)	$Decl(M) \triangleq T$	(16)
declaration of field F (declared in type T)	$Decl(F) \triangleq T$	(17)
explicit declaration of variable or method parameter $T\ v$	$[v] \triangleq T$	(18)
declaration of method M with return type T	$[M] \triangleq T$	(19)
declaration of field F with type T	$[F] \triangleq T$	(20)
cast $(T)E$	$[(T)E] \triangleq T$	(21)
expression new $C(E_1, \cdots, E_n)$	$[\text{new } C(E_1, \cdots, E_n)] \triangleq C$	(22)
M' overrides M, $M' \neq M$	$[Param(M', i)] = [Param(M,i)]$ $[M'] = [M]$	(23) (24)
for each cast expression $(C)E$, and each allocation expression $E' \in PointsTo(P,E)$ such that $[E']_P \leq [(C)E]_P$	$[E'] \leq [(C)E]$	(25)
for each cast expression $(C)E$, and each allocation expression $E' \in PointsTo(P,E)$ such that $[E']_P \not\leq [(C)E]_P$	$[E'] \not\leq [C) = [(C)E]$	(26)
expression E that occurs in the libraries such that $[E]_P = T$	$[E] = T$	(27)
allocation expression new $C(E_1, \cdots, E_n)$) in the libraries	$[\text{new } C(E_1, \cdots, E_n)] = C$	(28)

FIG. 5

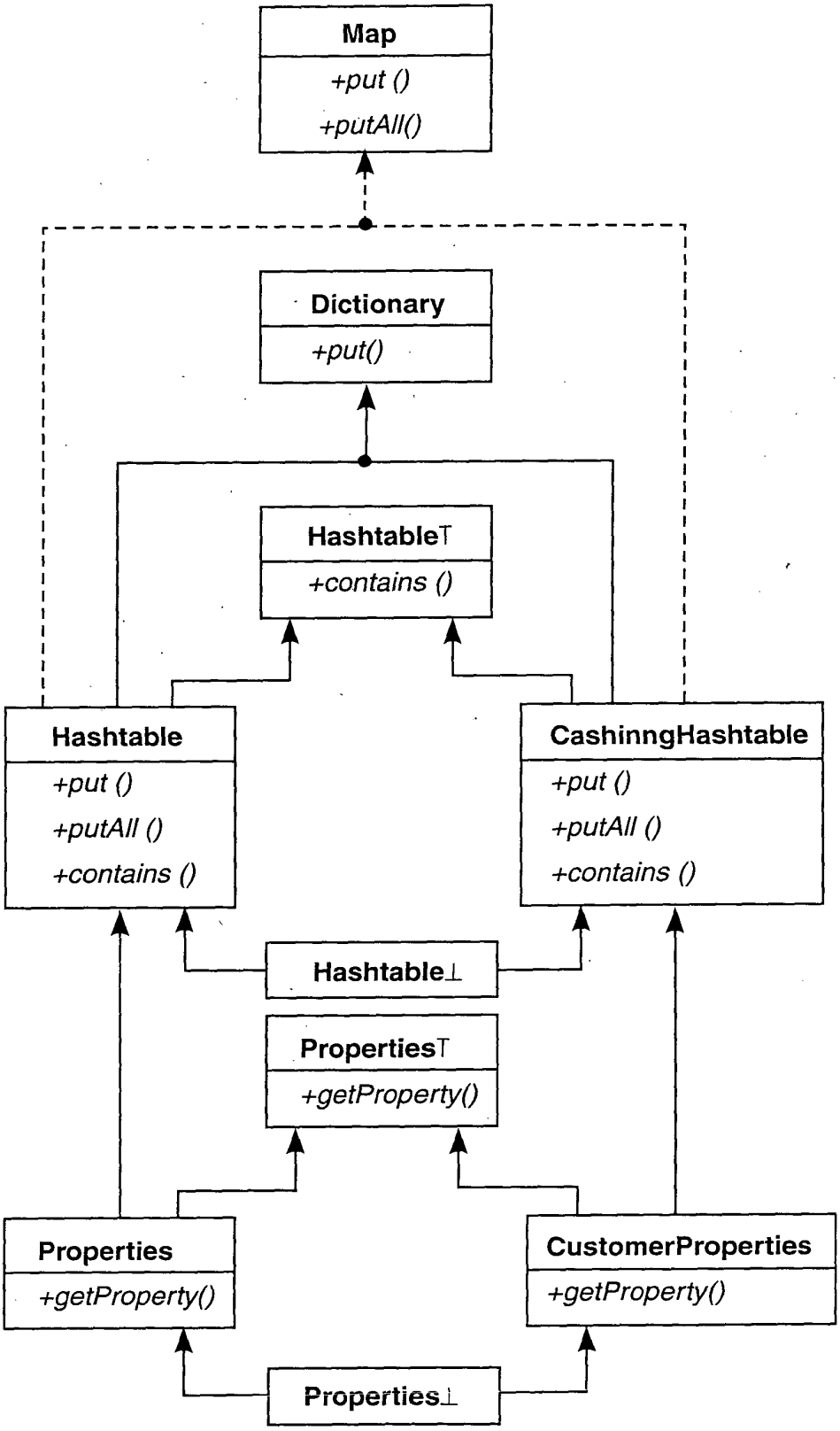
Map
+put ()
+putAll()
Dictionary
+put()
HashtableT
+contains ()
Hashtable
+put ()
+putAll ()
+contains ()
CashinngHashtable
+put ()
+putAll ()
+contains ()
Hashtable⊥
PropertiesT
+getProperty()
Properties
+getProperty()
CustomerProperties
+getProperty()
Properties⊥

FIG. 6

program construct(s) / analysis facts	implied type constraint(s)
expression new $C(E_1, \cdots, E_n)$ C is a customizable class	$[\text{new } C(E_1, \cdots, E_n)] \leq C^{\top}$ (22)(a) $C^{\perp} \leq [\text{new } C(E_1, \cdots, E_n)]$ (22)(b)
expression new $C(E_1, \cdots, E_n)$ C is a non-customizable class	$[\text{new } C(E_1, \cdots, E_n)] \triangleq C$ (22)(c)

FIG. 7

line	original constraint	rule
3	[H1] ≤ [h1]	(1)
3	[H1] ≤ Hashtable$^{\top}$	(22)(a)
3	Hashtable$^{\perp}$ ≤ [H1]	(22)(b)
4	[h1] ≤ [*Param*(JTree.JTree(),1)]	(8)
4	[*Param*(JTree.JTree(),1)] = Hashtable	(27)
5	[H2] ≤ [h2]	(1)
5	[H2] ≤ Hashtable$^{\top}$	(22)(a)
5	Hashtable$^{\perp}$ ≤ [H2]	(22)(b)
6	[H3] ≤ [h3]	(1)
6	[H3] ≤ Hashtable$^{\top}$	(22)(a)
6	Hashtable$^{\perp}$ ≤ [H3]	(22)(b)
7/15	[h3] ≤ [o]	(10)
8	[h3] ≤ [h2]	(1)
9	[h2] ≤ Map or [h2] ≤ Dictionary	(4)
10	[h2] ≤ Map	(4)
11	[System.getProperties()] ≤ [p1]	(1)
11	[System.getProperties()] = Properties	(27)
12	[p1] ≤ Properties	(4)
13	[p1] ≤ [h2]	(1)
16	[C1] ≤ [o]	(12)
16	[H3] ≤ [C1]	(25)
16	[S1] ≰ [C1]	(26)
16	[C1] ≤ [h4]	(1)
17	[h4] ≤ Hashtable$^{\top}$	(4)
20	S1 ≤ [s]	(1)
20	[S1] = String	(27)
21/15	[s] ≤ [o]	(10)

line	original constraint	rule
3	[H1] ≤ [h1]	(1)
3	[H1] ≤ Hashtable$^{\top}$	(22)(a)
3	Hashtable$^{\perp}$ ≤ [H1]	(22)(b)
4	[h1] ≤ [*Param*(JTree.JTree(),1)]	(8)
4	[*Param*(JTree.JTree(),1)] = Hashtable	(27)
5	[H2] ≤ [h2]	(1)
5	[H2] ≤ Hashtable$^{\top}$	(22)(a)
5	Hashtable$^{\perp}$ ≤ [H2]	(22)(b)
6	[H3] ≤ [h3]	(1)
6	[H3] ≤ Hashtable$^{\top}$	(22)(a)
6	Hashtable$^{\perp}$ ≤ [H3]	(22)(b)
7/15	[h3] ≤ [o]	(10)
8	[h3] ≤ [h2]	(1)
10	[h2] ≤ Map	(4)
11	[System.getProperties()] ≤ [p1]	(1)
11	[System.getProperties()] = Properties	(27)
12	[p1] ≤ Properties	(4)
13	[p1] ≤ [h2]	(1)
16	[C1] ≤ [o]	(12)
16	[H3] ≤ [C1]	(25)
16	[C1] ≤ Hashtable$^{\top}$	(26)
16	[C1] ≤ [h4]	(1)
17	[h4] ≤ Hashtable$^{\top}$	(4)
20	S1 ≤ [s]	(1)
20	[S1] = String	(27)
21/15	[s] ≤ [o]	(10)

FIG. 8

equivalence class	possible types
{p1, Properties }	{Properties }
{h1 }	{Hashtable }
{H1 }	{Hashtable }
{h2 }	{Map, Hashtable, CustomHashtable }
{H2 }	{Hashtable, CustomHashtable }
{h3 }	{Map, Hashtable, CustomHashtable }
{H3 }	{Hashtable, CustomHashtable }
{h4 }	{Hashtable, CustomHashtable }
{C1 }	{Hashtable, CustomHashtable }
{o }	{Object }
{s }	{String }

FIG. 9

	Sun JVM 1.3.1					j9				
	time (sec)			ratios		time (sec)			ratios	
benchmark	P_o	P_u	P_c	P_u/P_o	P_c/P_o	P_o	P_u	P_c	P_u/P_o	P_c/P_o
_202_jess	67.3	65.2	62.1	0.97	**0.92**	51.5	51.3	48.6	1.00	**0.94**
_209_db	79.4	78.5	65.3	0.99	**0.82**	68.7	66.6	67.8	0.97	**0.99**
_218_jack	83.4	83.3	76.5	1.00	**0.92**	60.0	62.5	54.7	1.04	**0.91**
hyperJ	22.5	23.7	20.7	1.05	**0.92**	23.7	25.5	22.1	1.08	**0.93**
Jax	25.7	25.7	24.9	1.00	**0.97**	33.3	33.5	33.2	1.00	**1.00**
PmD	6.93	6.88	6.53	0.99	**0.94**	5.46	5.42	5.15	0.99	**0.94**

FIG. 10

benchmark	# alloc.	customizations
_202_jess	1 HT	F, NI, KS, SE
	1 HT	F, NI, KS, AB
	1 HT	F, NI, KI
	1 HT	F, NI, KS
_209_db	1K V	F, NI, SCL
_228_jack	10 HT	F, NI, AS, SEO
	1K HT	F, NI, AL
	1K HT	F, NI, AL
HyperJ	1 HT	F, SCL
	10K HT	NI, AL, SEO
	10K HT	F, NI, AL
Jax	10K HT	F, NI, AS
	3K HT	F, NI, AS
PmD	10K HM	F, AL
	100K HS	F, NI, AL, SEO
	20K HS	F, NI, AL, SEO

FIG. 11

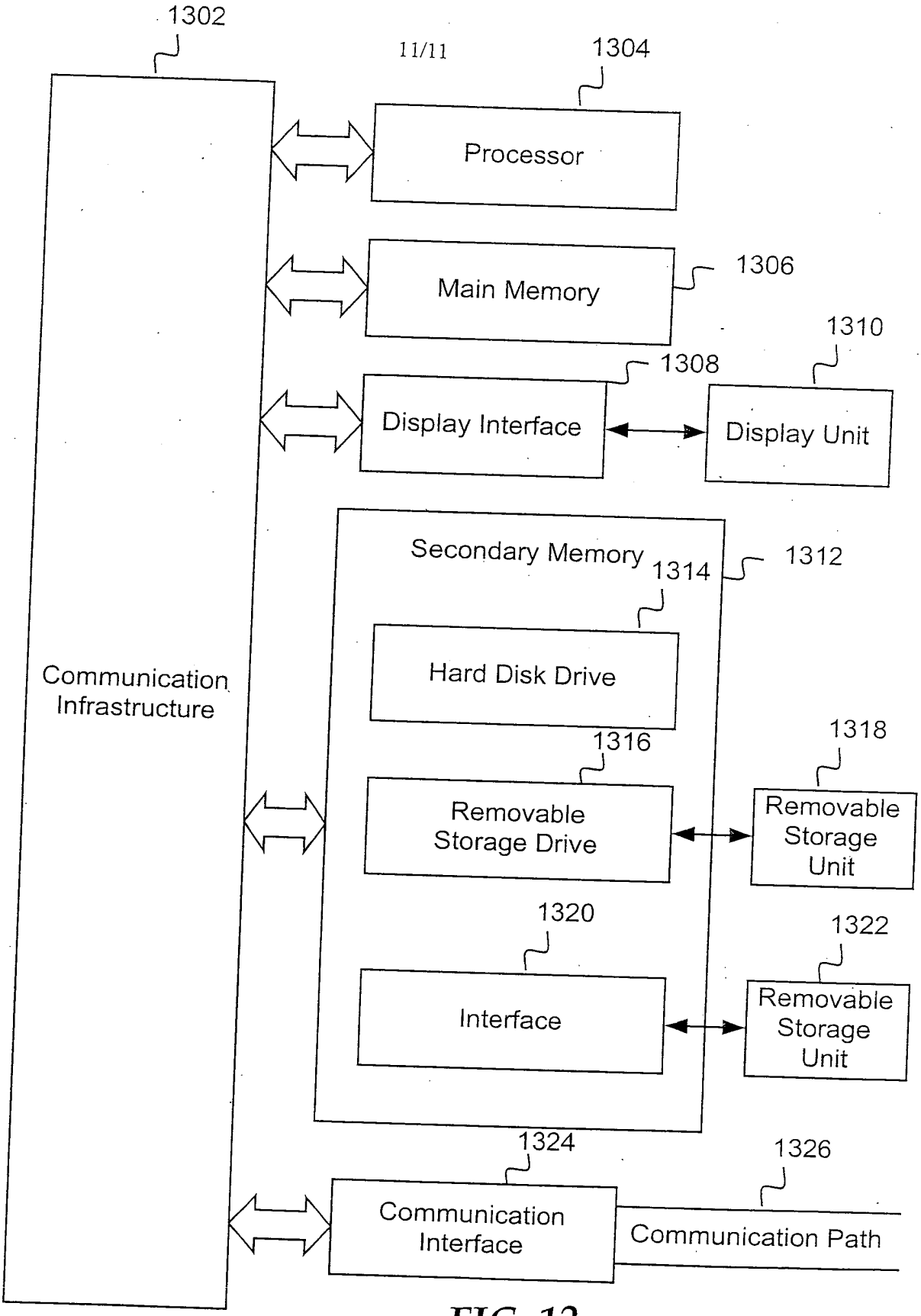
1302
1304
Processor
Main Memory
1306
1310
1308
Display Interface
Display Unit
Secondary Memory
1312
1314
Hard Disk Drive
Communication Infrastructure
1316
1318
Removable Storage Drive
Removable Storage Unit
1320
1322
Interface
Removable Storage Unit
1324
1326
Communication Interface
Communication Path

FIG. 12